BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK-AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

TITLE: VIDEO SPLITTING AND DISTRIBUTED PLACEMENT SCHEME FOR ...
INVENTORS: Hai JIN et al.
APPLICATION NO.: New
DOCKET NO.: 1793.1189

FIG. 1

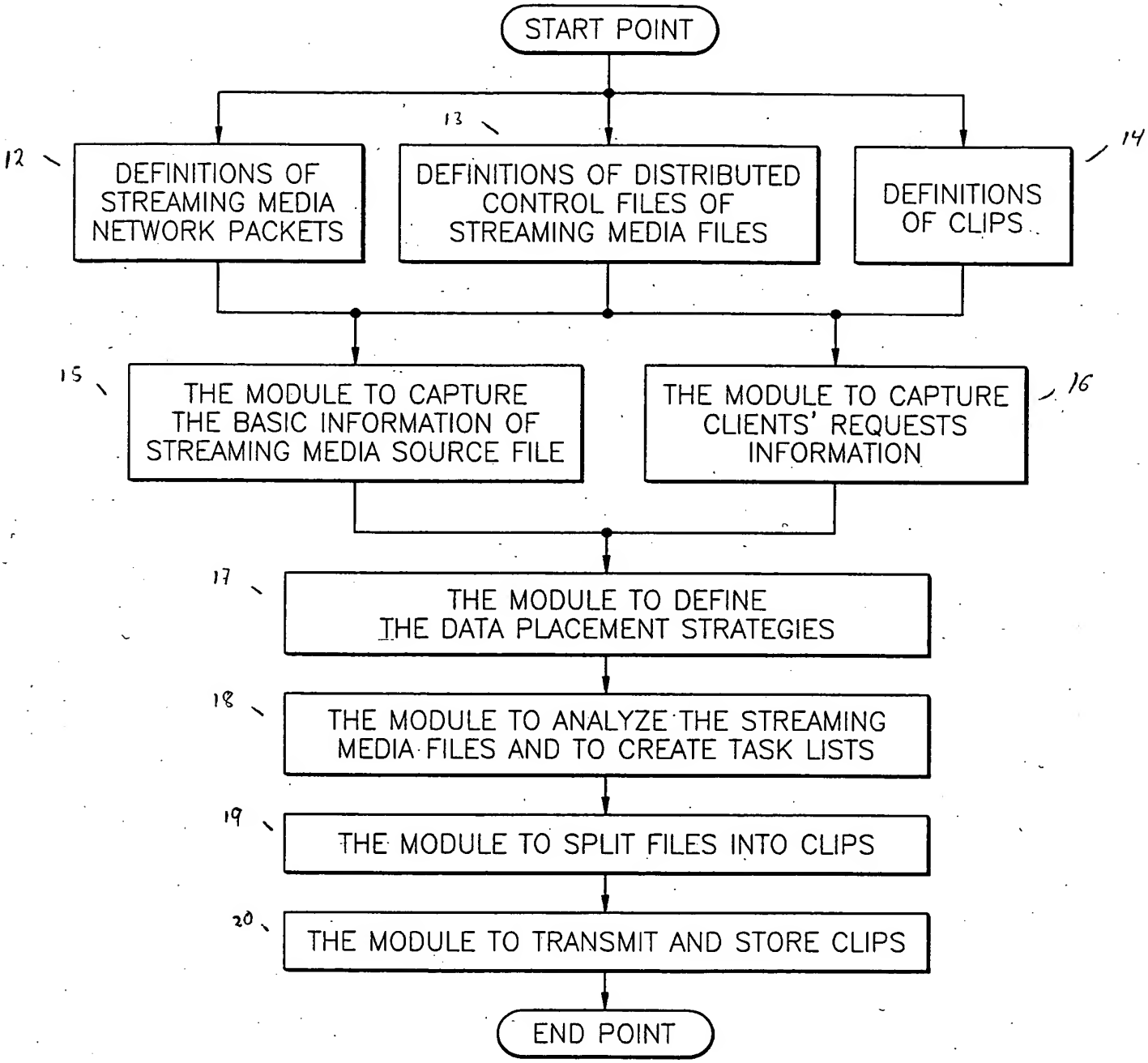


FIG. 2

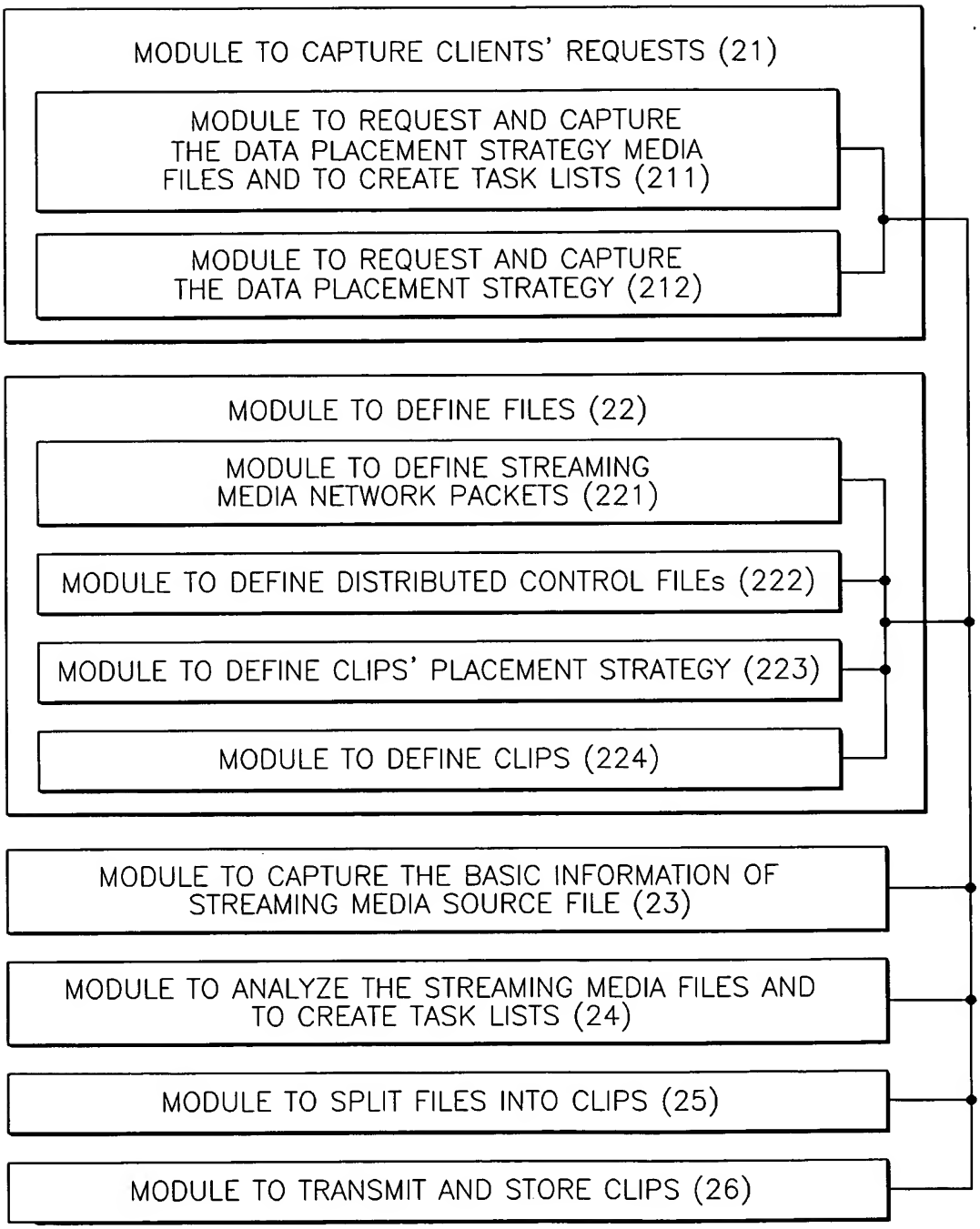
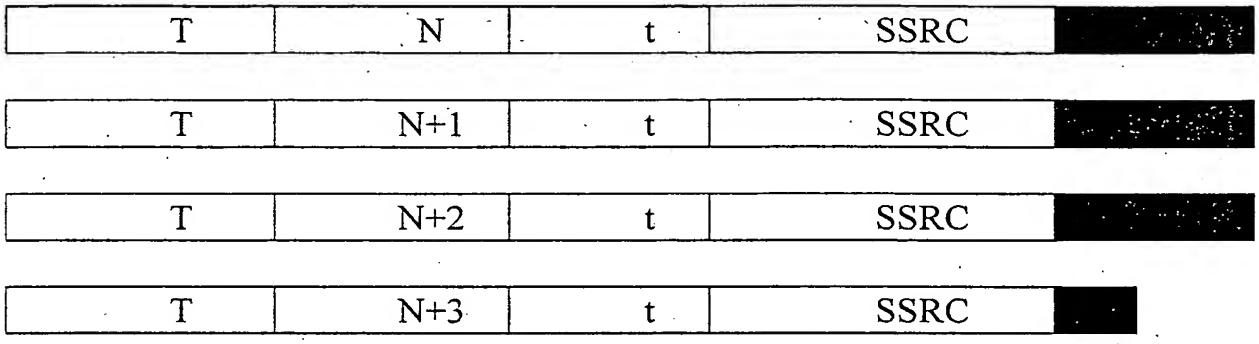

MODULE TO CAPTURE CLIENTS' REQUESTS (21)
MODULE TO REQUEST AND CAPTURE THE DATA PLACEMENT STRATEGY MEDIA FILES AND TO CREATE TASK LISTS (211)
MODULE TO REQUEST AND CAPTURE THE DATA PLACEMENT STRATEGY (212)
MODULE TO DEFINE FILES (22)
MODULE TO DEFINE STREAMING MEDIA NETWORK PACKETS (221)
MODULE TO DEFINE DISTRIBUTED CONTROL FILEs (222)
MODULE TO DEFINE CLIPS' PLACEMENT STRATEGY (223)
MODULE TO DEFINE CLIPS (224)
MODULE TO CAPTURE THE BASIC INFORMATION OF STREAMING MEDIA SOURCE FILE (23)
MODULE TO ANALYZE THE STREAMING MEDIA FILES AND TO CREATE TASK LISTS (24)
MODULE TO SPLIT FILES INTO CLIPS (25)
MODULE TO TRANSMIT AND STORE CLIPS (26)

FIG. 3

STREAMING MEDIA NETWORK PACKETS:

MEDIA TYPE HEAD
SEQUENCE NUMBER TIME STAMP SYNCHRONOUS SOURCE PAYLOAD



MEDIA TYPE HEAD: UNSIGNED SHORT INTEGER WITH 16 BITS
SEQUENCE NUMBER: UNSIGNED SHORT INTEGER WITH 16 BITS
TIME STAMP: UNSIGNED INTEGER WITH 32 BITS
SYNCHRONOUS SOURCE: UNSIGNED INTEGER WITH 32 BITS
PAYLOAD: AN ARRAY CONSISTED OF UNSIGNED CHAR WITH 8 BITS

FIG. 4

```
//the clips' information
typedef struct Clip
{
    //the storage addresses set of the clips and replicas
    Unsigned int (32 bits)          fHost IP [ MAX_REPLICA_NUM ] ;
    //the space size of the clip.
    Unsigned int 32 bits            fFileSize;
    //the start playtime counted in seconds;
    Float 64 bits                   fStartTime;
    //the end playtime counted in seconds;
    Float 64 bits                   fEndTime;
    //the sequence number of the first network packet of the clip;
    Unsigned int 32 bits            fStartPacketIndex;
    //the sequence number of the last network packet of the clip;
    Unsigned int 32 bits            fEndPacketIndex;
} Clip;

//the splitting task lists of one source file
typedef struct ClipTable
{
    //the space size of one media file;
    Unsigned int (32 bits)      fFileSize;
    //the hot optionof one film;
    int                         fHot;
    //the name length of the source media file;
    Unsigned char (8 bits)              fNameLen;
    //the number of clips of one source media file;
    Unsigned char (8 bits)              fNumber;
    //the name of the source media file;
    Char                        *fName
    //the structure of each list item;
    Clip                        *fIndex;
} ClipTable;
```

FIG. 5

```
b=AS:1383
a=range:npt=0-   46.57500
m-OTHER 0 RTP/AVP 96
b=AS:1383
a=rtpmap:96 MP1S/90000
a=control:trackID=2
```

FIG. 6

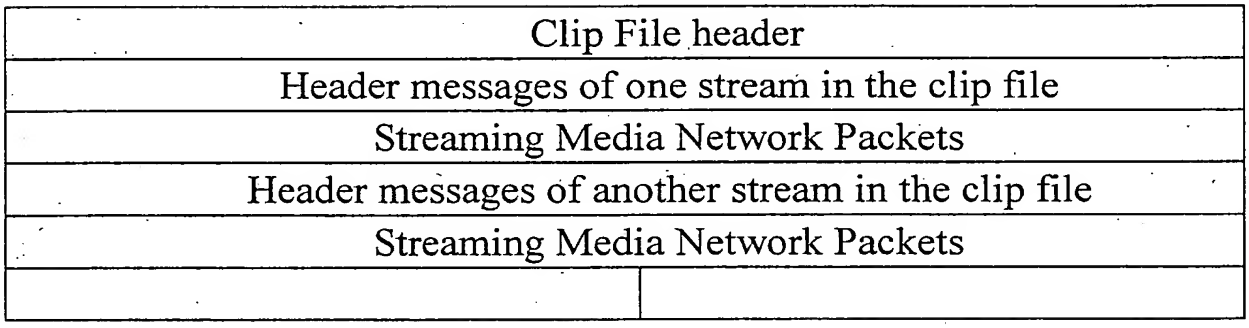


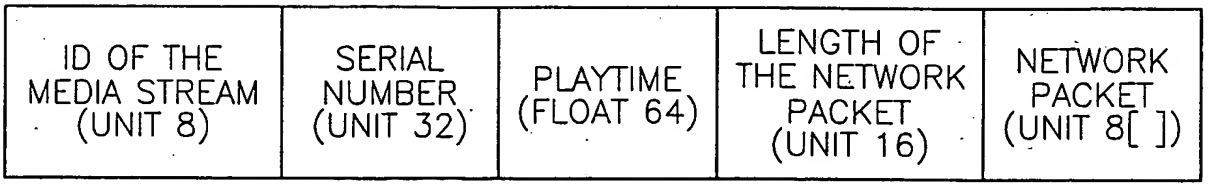
FIG. 7

```
typedef struct FileHeader
{
    // Index ID of one clip file
    Unsigned int (16 bits)          fSplit_ID;
    //The version of the current splitting tool
    Unsigned int (32 bits)          fVersion;
    //The time length of the clip
    Float 64 (64 bits)    fMovieDuration
    //The number of the media streams in the clip
    Unsigned int (32 bits)          fNumTracks
    //The average bandwidth of the clip
    Float 64 (64 bits)    fBandWidth;
} File Header
```

FIG. 8

```
typedef struct TrackHeader
{
    //ID of the stream
    Unsigned char (8 bits)              fTrackID;
    //The duration of the stream
    Float (64 bits)          fTrackDuration;
    //The compression ratio of the stream
    Float (64 bits)          fCompressRatio;
    //The start location of the media data of the stream
    Unsigned int (32 bits)          fMediaPosition
} Fileheader
```

FIG. 9



UINT8: UNSIGNED CHAR (8 BITS)
UINT16: UNSIGNED INT (16 BITS)
UINT32: UNSIGNED INT (32 BITS)
FLOAT64: FLOAT (64 BITS)

FIG. 10

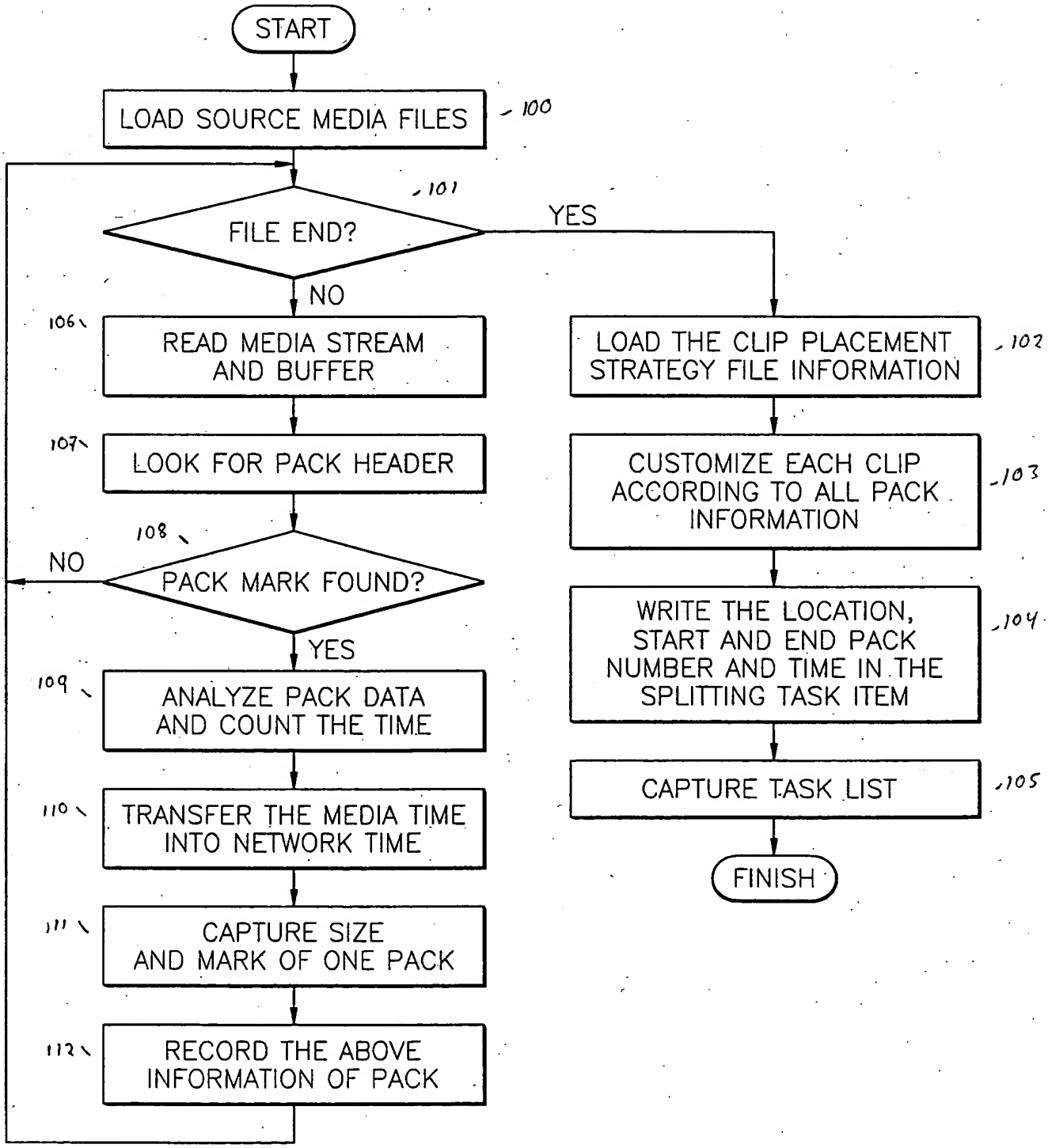

START
LOAD SOURCE MEDIA FILES
100
FILE END?
101
YES
NO
READ MEDIA STREAM AND BUFFER
106
LOOK FOR PACK HEADER
107
PACK MARK FOUND?
108
NO
YES
ANALYZE PACK DATA AND COUNT THE TIME
109
TRANSFER THE MEDIA TIME INTO NETWORK TIME
110
CAPTURE SIZE AND MARK OF ONE PACK
111
RECORD THE ABOVE INFORMATION OF PACK
112
LOAD THE CLIP PLACEMENT STRATEGY FILE INFORMATION
102
CUSTOMIZE EACH CLIP ACCORDING TO ALL PACK INFORMATION
103
WRITE THE LOCATION, START AND END PACK NUMBER AND TIME IN THE SPLITTING TASK ITEM
104
CAPTURE TASK LIST
105
FINISH

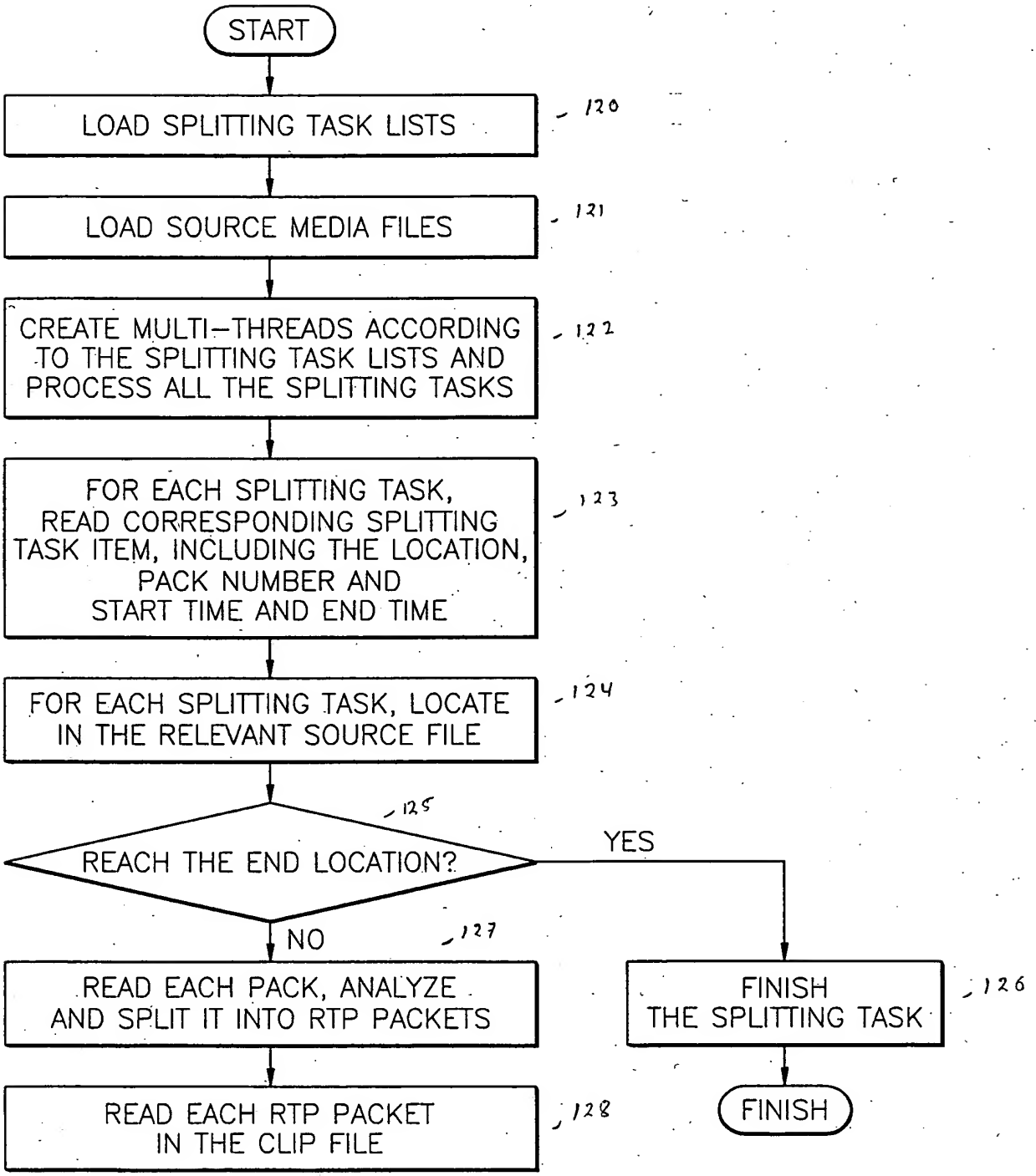
FIG. 11

```
typedef struct Each_Task_Info
    {
    //start time of the clip counted in seconds;
    Float (64 bits)          fStartTime;
    //start Pack sequence number of the clip;
    Unsigned int (32 bits)       fStartPackIndex;
    //Sstart offset of the clip
    Unsigned int (32 bits)       fStartPosition;
    // end offset of the clip
    Unsigned int (32 bits)       fEndPosition;
    // index of the clip
    Unsigned int (32 bits)       fIndex;
    //task finish percentage of the clip;
    Float (64 bits)          fWorkingProcessing;
    //task dispatch start time of the clip
    Time             fSchedule_Start_Time;
    //total time of finishing the task of the clip. Its unit is second;
    Float (64 bits)          fSchedule_Total_Time;
    //whether the task of the clip is successful or not
    Unsigned char (8 bits)           fSucceed;
    } Each_Task_Info;

typedef struct Task_Info
{
    //the number of the items in the list
    Unsigned char (8 bits)           fNumber;
    //the handle of the source media file in the list
    int                  fSourceFile
    //all the splitting tasks
    Each_Task_Info *fIndex [ MAX_SPLIT_NUMBER ] ;

} Task_Info;
```

FIG. 12


START
LOAD SPLITTING TASK LISTS
120
LOAD SOURCE MEDIA FILES
121
CREATE MULTI-THREADS ACCORDING TO THE SPLITTING TASK LISTS AND PROCESS ALL THE SPLITTING TASKS
122
FOR EACH SPLITTING TASK, READ CORRESPONDING SPLITTING TASK ITEM, INCLUDING THE LOCATION, PACK NUMBER AND START TIME AND END TIME
123
FOR EACH SPLITTING TASK, LOCATE IN THE RELEVANT SOURCE FILE
124
REACH THE END LOCATION?
125
YES
NO
127
READ EACH PACK, ANALYZE AND SPLIT IT INTO RTP PACKETS
READ EACH RTP PACKET IN THE CLIP FILE
128
FINISH THE SPLITTING TASK
126
FINISH